Controller Driven VRML Animation of the Next Generation Inspection System (NGIS) Real-time Controller

Keith Stouffer*, John Horst
National Institute of Standards and Technology
100 Bureau Drive, Stop 823, Gaithersburg, MD 20899-8230

ABSTRACT

Virtual objects in a web-based environment can be interfaced to and controlled by external real world controllers. A Virtual Reality Modeling Language (VRML) inspection cell was created that models a real-time inspection system. The tested consists of a Cordax Coordinate Measuring Machine (CMM), a vision system for determining the part position and orientation, and an open architecture controller. Because of the open architecture, data such as the probe position and the part position and orientation, can be obtained from the controller to drive a VRML model of the system. The VRML CMM is driven using a socket connection between the collaborator's web browser and the real world controller. The current probe position, which is stored in a world model buffer in the controller, is collected by a Java applet running on the web page. The applet updates the VRML model of the CMM via the External Authoring Interface (EAI) of the VRML plug-in. The part position and orientation is obtained from the vision system and the part is updated in the VRML model to represent the part's real world position and orientation. The remote access web site also contains a client-controlled pan/tilt/zoom camera, which sends video to the client allowing them to monitor a remote inspection with a PC and an Internet connection.

Keywords: online monitoring; open architecture control; remote access and inspection; VRML

1. INTRODUCTION

The Intelligent Systems Division maintains an inspection testbed consisting of a Cordax Coordinate Measuring Machine (CMM), advanced sensors, and the National Institute of Standards and Technology (NIST) Real-Time Control System (RCS) open architecture controller. The goals of this work are, in part, to implement and test open architecture concepts in a metrology system, to develop and demonstrate feature-based inspection, and to develop and implement a fixtureless inspection subsystem. The specific goal of the NGIS project, which contributed substantually to this testbed, is to increase the speed and flexibilty of data acquisition using CMMs while still maintaining today's accuracy. The NGIS project involves a consortium of companies organized by the National Center for Manufacturing Sciences.

A Virtual Reality Modeling Language (VRML) inspection cell was created that models the real-time inspection system. The inspection system consists of the Cordax CMM, an inspection probe, a vision system for determining the part position and orientation, and an open architecture controller.[1.] The contact probe and the vision system camera are mounted on the CMM arm. The controller sends velocity commands to each of the three axis motors every 5 ms and it reads each of the three axis positions every 2 ms. Figure 1 on the next page shows the CMM arm, the part to be inspected, the camera mounted on the arm, and the contact probe. The controller performs real-time processing of sensor data for feedback control of the inspection probe. The open architecture controller permits access to data, such as the probe position and the part position and orientation, to drive a VRML model of the system.

In order to inspect a part, the operator specifies the features that need to be measured. An inspection plan is automatically generated from the computer aided design (CAD) solid model of the part. The inspection plan is then translated into Dimensional Measurement Interface Standard (DMIS) code. An interpreter converts the DMIS code into conanical control commands for the CMM and vision subsystems.[2.] The CMM is then commanded to move to a predefined position and the part to be measured is placed on the CMM table in an arbitrary position and orientation. The operator signals that the part is on the table and the system determines the position and orientation (i.e. pose) of the part using the camera and computer vision algorithms. The inspection plan is then executed.

* Correspondence: Email: stouffer@cme.nist.gov; WWW: http://www.isd.mel.nist.gov/personnel/stouffer;
Telephone: 301 975 3877; FAX: 301 990 9688

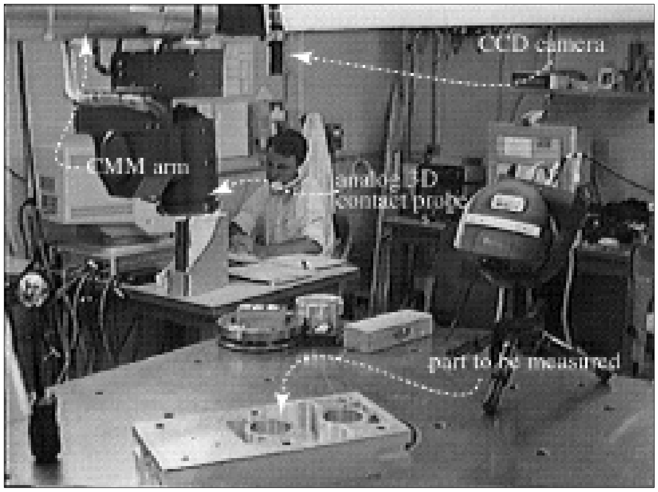

Figure 1. The Inspection System

2. REMOTE ACCESS PAGE

Figure 2 shows the web-based remote access web site that was developed. The page contains a VRML model of the inspection cell that is controlled by the real world controller located at NIST and a collaborator-controlled pan/tilt/zoom camera that sends video to the collaborator. This remote access web site allows a collaborator to monitor a remote inspection operation with a PC and an Internet connection.

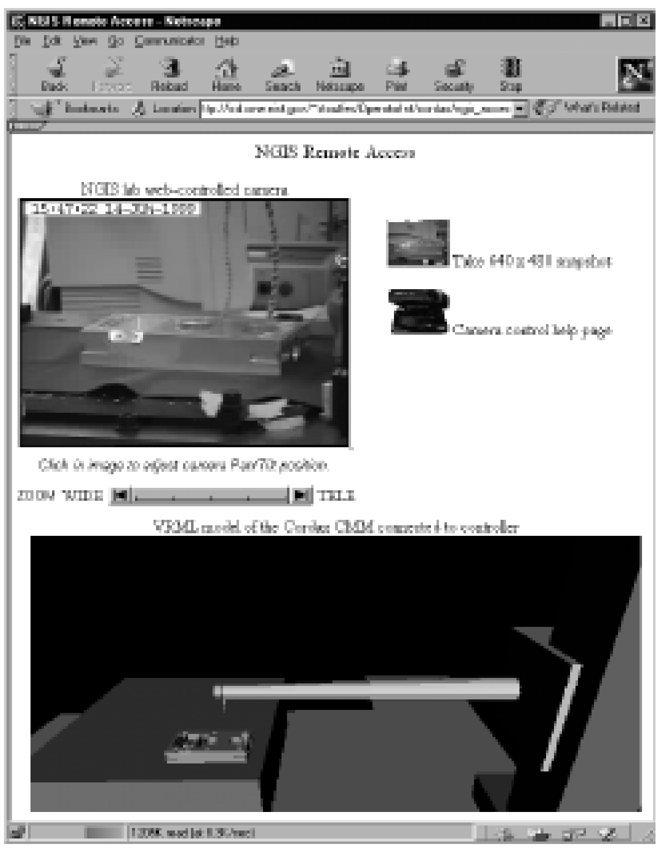

Figure 2. Remote access page with the client controllable pan/tilt/zoom camera and controller driven VRML model of the inspection cell.

2.1 VRML model of the inspection system

The CMM was originally modeled using a robotic simulation package from Deneb Robotics called TeleGRIP. This simulation package allows for design, evaluation, and off-line programming of robotic workcells, incorporating real world robotic and peripheral equipment, motion attributes, kinematics, dynamics and I/O logic.

In order to view the 3D model of the CMM within a web browser, the model must first be translated to VRML. A translator program was written by Qiming Wang and Sandy Ressler of the Information Technlogy Laboratory (ITL) at NIST which can be used to translate devices and workcells generated by TeleGRIP to VRML.[3.] The translation includes the geometry information as well as the kinematic information of the device. Figure 3 shows the translated VRML model viewed within a web browser using a VRML plug-in such as Cosmoplayer or Blaxxun Contact.

The real world inspection controller drives the VRML CMM. This is accomplished by a socket connection between the collaborator's web browser and the real world controller. When a client visits the remote access web page (Figure 2), a VRML model of the inspection cell is downloaded to his or her machine. Once the VRML model of the inspection cell has been downloaded to the client's machine, only the current joint positions need to be sent across the socket from the controller (server) to the web browser (client) to move the joints of the VRML CMM model to the current controller position. All of the graphics are handled on the client machine.

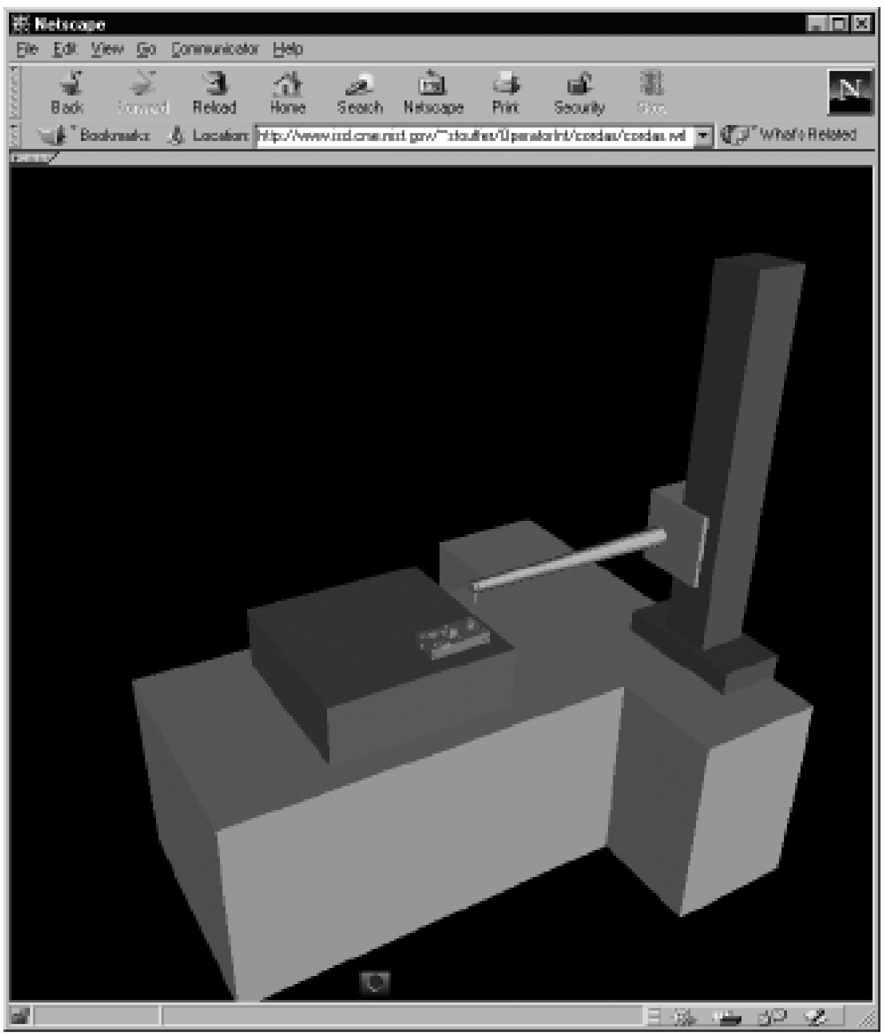

Figure 3. VRML model of the inspection system including the part to be measured

2.2 Pan/Tilt/Zoom remote camera system

The remote access web site also contains a client-controlled pan/tilt/zoom camera which sends video to the client. This allows a client to monitor a remote inspection with a PC and an internet connection. The video within the remote access page is a server-pushed JPEG image with an imagemap overlaid on it. The resolution of each image is 320 x 240 pixels. Each image is fairly large (20 Kbytes), therefore, frame rates are slow (1 frame per second). Each image of the video has an overlaid imagemap, allowing the collaborator to click on the image where the new center of interest should be. The required pan/tilt angles are calculated and the camera moves to recenter the image. The zoom of the camera is controlled by a scroll bar with WIDE (zoom out) and TELE (zoom in) at the ends of the scrollbar.

3. VISION SYSTEM

Before a part can be inspected, the vision system must determine the part's position and orientation, and then calculate the coordinate transformation between the part coordinate system and the machine coordinate system. The vision system automates this process during setup using monocular vision for parts with 2D features. The automated setup algorithm consists of an image processing algorithm and a pose estimation algorithm. The image processing algorithm, called Lola,[4.] produces line segment features and constant curvature arc features. The pose estimation algorithm matches sensed with model features and performs pose clustering and pose verification. The pose estimation is shown in Figure 4. The black features are the sensed features and the gray features are model features translated and rotated by the computed pose estimate. Note that a correct match is made in spite of warped, occluded, and spurious sensed features.

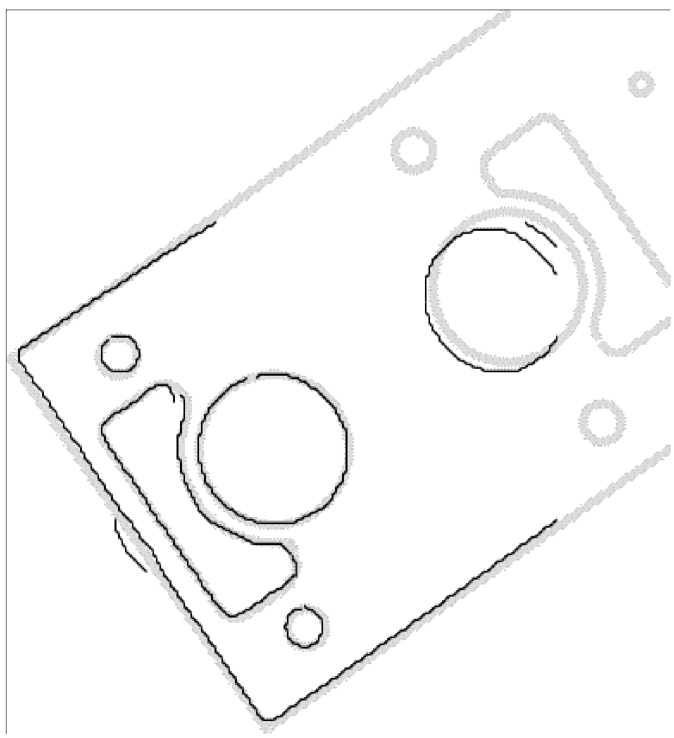

Figure 4. Part pose estimation

The matching segment relates the problem of finding a correspondence between sensed and model features to that of finding a word in a dictionary. A dictionary of words is formed where the letters consist of translation and rotation invariant geometric attributes of all possible sets of k model features. Each word in the dictionary is sorted by canonical order of the letters within the word and the entire dictionary is sorted according to the canonical order of the words. At run-time, $k < s$ sensed features are randomly selected from s sensed features. The sensed word is formed and all matches between sensed and model words are found. The part position and orientation is obtained from the vision system and updated in the VRML model to represent the part's real world position and orientation.

4. CONTROLLING THE VRML MODEL

The controller driven VRML model of the inspection cell augments the capabilities of the camera system by giving collaborators a 3D perspective of cell motions and events with faster update rates. The VRML robot is interfaced to and controlled by the real world robot controller. This is accomplished by a Neutral Messaging Language (NML)[5.] socket connection between the collaborator's web browser and the real world controller. This socket is created by a Java applet running on the remote access page. The current joint angles of the robot, which are stored in a world model buffer in the controller, are collected by the Java applet. In general, the communications between an NML client and server are fairly simple. The server starts, opens a configuration file,[6.] reads whether to use Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) and a port number, and then binds to the port and listens for requests. Then a client starts and reads the same configuration file (which tells the port whether to use TCP or UDP, and on which host the server is running), connects to the appropriate port, and sends requests to the server. The client can send requests to write a message, read a message only one-time, or read a message and automatically be sent updates whenever the data changes or at a particular rate.

4.1 TCP proxy

There are several obstacles to running controller driven VRML animations within a web page. Web browsers usually prevent Java applets running inside them from connecting to any hosts other than the web server from which they were loaded. Many institutions also have a firewall in place that prevents any host outside the wall from directly contacting hosts inside the wall. One way to circumvent these problems is to run a TCP proxy. A TCP proxy is a generic program to forward TCP data from one host to another to provide a very limited bypass of these security restrictions.

The TCP proxy is started on the web server with command line arguments indicating the host with the NML server and the TCP port number. It binds that port on the web server and listens for connections. Each time a client connects to the TCP proxy, the TCP proxy connects to the NML server on the controller. It forwards any requests made by that client to the server and any data sent by the server back to the client in such a way that neither the client nor the server is aware that a TCP proxy is involved at all.

4.2 Low bandwidth connection

When a collaborator visits the web page, a VRML model of the inspection cell is downloaded to his or her machine. Once the VRML model has been downloaded to the client's machine, only the joint positions need to be sent across the socket from the controller (server) to the web browser (client) to move the joints of the VRML robot to the current controller position. All the graphics of redrawing the robot position are handled on the client machine. The data that is sent across the socket consists of 3 floats (x, y, and z position of the CMM probe) converted to the eXternal Data Representation (XDR), so the format is not processor specific. This data (3 floats = 12 bytes) is sent across the socket every 30 ms, giving a data rate of 400 bytes/sec. This low bandwidth is usually handled easily, even on a slow network connection.

4.3 External authoring interface (EAI)

Once the data has been gathered from the controller by the Java applet, the applet must communicate with the VRML CMM to animate it. For communication between a VRML world and an external environment (the applet), an interface between the two is needed. A proposed interface, called the External Authoring Interface (EAI),[7.] defines the set of functions on the VRML plug-in that the external environment can perform to affect the VRML world. In essence, the EAI provides a method for developing custom applications that interact with, and dynamically update a VRML scene. The EAI allows an external program (the applet) to access nodes in a VRML scene using the existing VRML event model. The EAI Specification has been submitted to the VRML Review Board (VRB)[8.] for ISO formalization. Cosmoplayer 2.1 and Blaxxun Contact are two VRML plug-ins that incorporate the EAI.

Nodes in VRML can be named using the DEF construct. Any node named with the DEF construct can be accessed by the applet and is referred to as an accessible node. Once a pointer is obtained, the events (eventIns and eventOuts) of that node can be accessed. A Java applet communicates with a VRML world by first obtaining an instance of the `Browser` class. This class is the Java encapsulation of the VRML world. It contains the entire `Browser` Script Interface, as well as the `getNode()` method, which returns a `Node` when given a DEF name string. Only DEF names in the base VRML file are accessible. Names in Inline files and those created with `createVRMLFromString()` or `createVRMLFromURL()` are not accessible.

Once a `Node` instance is obtained from the `getNode()` method of the `Browser` class, its eventIns and eventOuts can be accessed. The `getEventIn()` method of the `Node` class returns an eventIn when passed a string with the desired eventIn name. The `getEventOut()` method of the `Node` class returns an eventOut when passed a string with the desired eventOut name. ExposedFields can also be accessed, either by giving a string for the exposedField itself (such as rotation) or by giving the name of the corresponding eventIn or eventOut (`set_rotation()` for the eventIn and `rotation_changed()` for the eventOut). A VRML joint class was developed to access and manipulate the VRML joints (each a separate DEFed Node).

4.4 Dynamic update of VRML part position and orientation

Before a part can be inspected, the vision system must determine the part's position and orientation, and then calculate the coordinate transformation between the part coordinate system and the machine coordinate system. After the part position and orientation is obtained from the vision system, they are updated in the VRML model via the EAI of the VRML plug-in to represent the part's real world position and orientation.

5. CONCLUSIONS

Virtual objects in a web-based environment can be interfaced to and controlled by external real world controllers. This allows animation of actual processes for collaborative verification. A VRML environment was created that models an inspection cell. The VRML inspection cell contains a model of a CMM and inspection part. The VRML CMM is interfaced to, and is controlled by, the real world controller. This is accomplished by a socket connection between the client's web browser and the real world controller. The current joint angles of the CMM, which are stored in a world model buffer in the controller, are collected by a Java applet running on the web page. The applet then updates the VRML model of the CMM via the EAI of the VRML plug-in. The part position and orientation is obtained from the vision system and the part is updated in the VRML model to represent the part's real world position and orientation. The web page also contains a client-controlled pan/tilt/zoom camera that sends video to the client allowing them to monitor a remote inspection with a PC and an Internet connection.

ACKNOWLEDGEMENTS

We would like to thank the SIMA program for the continuing support of this work, as well as Bill Rippey and Tommy Chang of the Intelligent Systems Division of NIST and David Jiang of the Advanced Technology and Research Corporation for substantial technical efforts. Without this support, our work would not have been possible.

DISCLAIMER

Any mention of commercial products within this paper is for information only; it does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products indentified are necessarily the best available for the purpose.

REFERENCES

1. Horst, John, *"Architecture, Design Methodology, and Component-Based Tools for a Real-Time Inspection System"*, The 3rd IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC 2000), Newport Beach, CA, March 15-17, 2000.

2. Messina, E., Horst, J., Kramer, T., Huang, H., Tsai, T., Amatucci, E., "A Knowledge-Based Inspection Workstation," Proceedings of the 1999 IEEE International Conference on Information, Intelligence and Systems.

3. Translator program from Deneb Robotics format to VRML: http://ovrt.nist.gov/projects/mfg/SIMA/deneb2vrml/deneb2vrml.html

4. Sarkar, S. and Boyer, K. L., "*Perceptual Organization in Computer Vision: A Review and a proposal for a Classificatory Structure*," IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 2, pp. 382-399, Mar. 1993.

5. Shackleford, W.P., "The NML Java Programmer's Guide", http://www.isd.mel.nist.gov/projects/rcs_lib/NMLjava.html

6. Shackleford, W.P., "Writing NML Configuration Files", http://www.isd.mel.nist.gov/proj/rcs_lib/NMLcfg.html

7. External Authoring Interface Working Group: http://www.vrml.org/WorkingGroups/vrml-eai/

8. VRML Review Board : http://www.vrml.org/vrb/